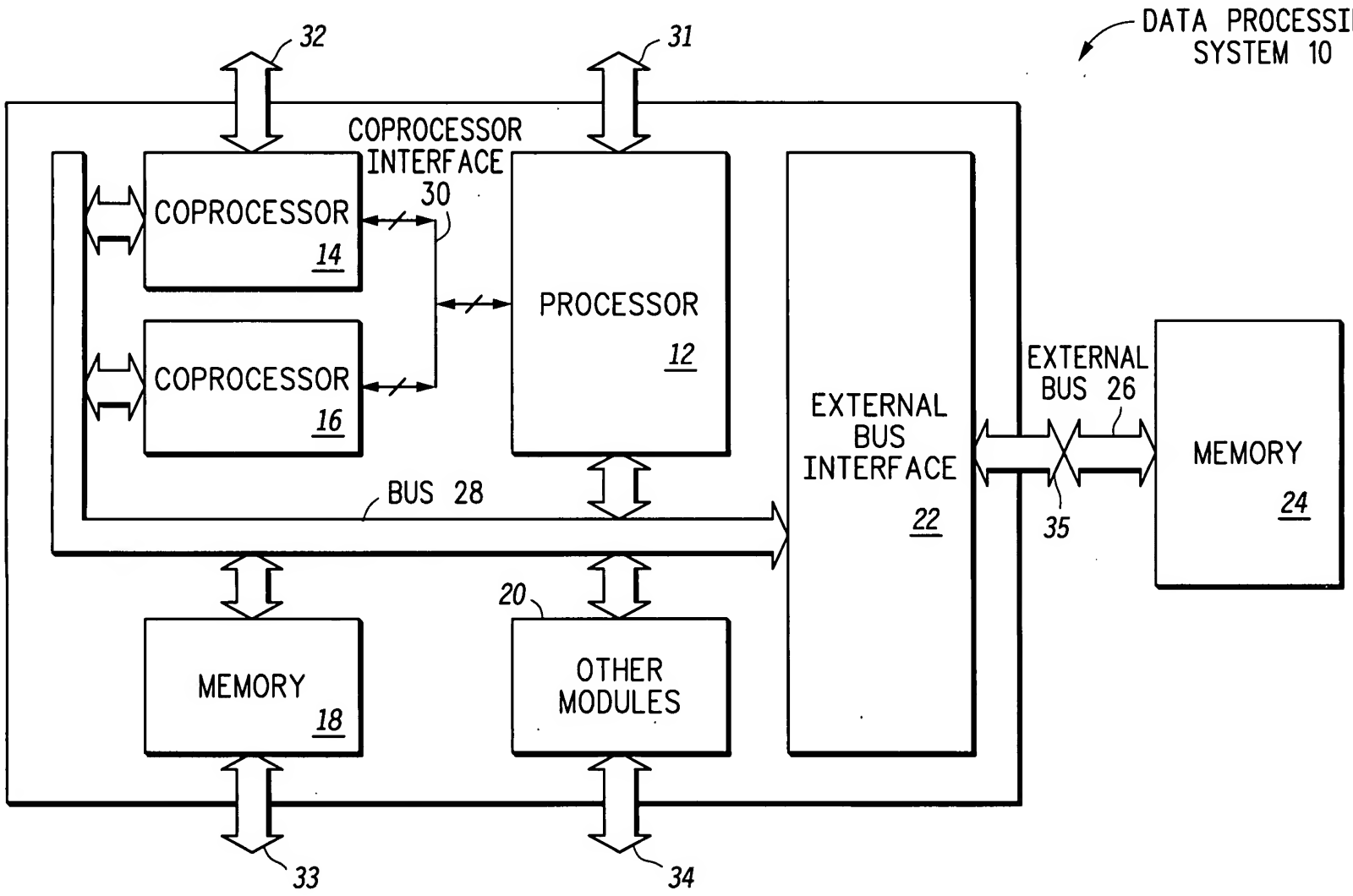
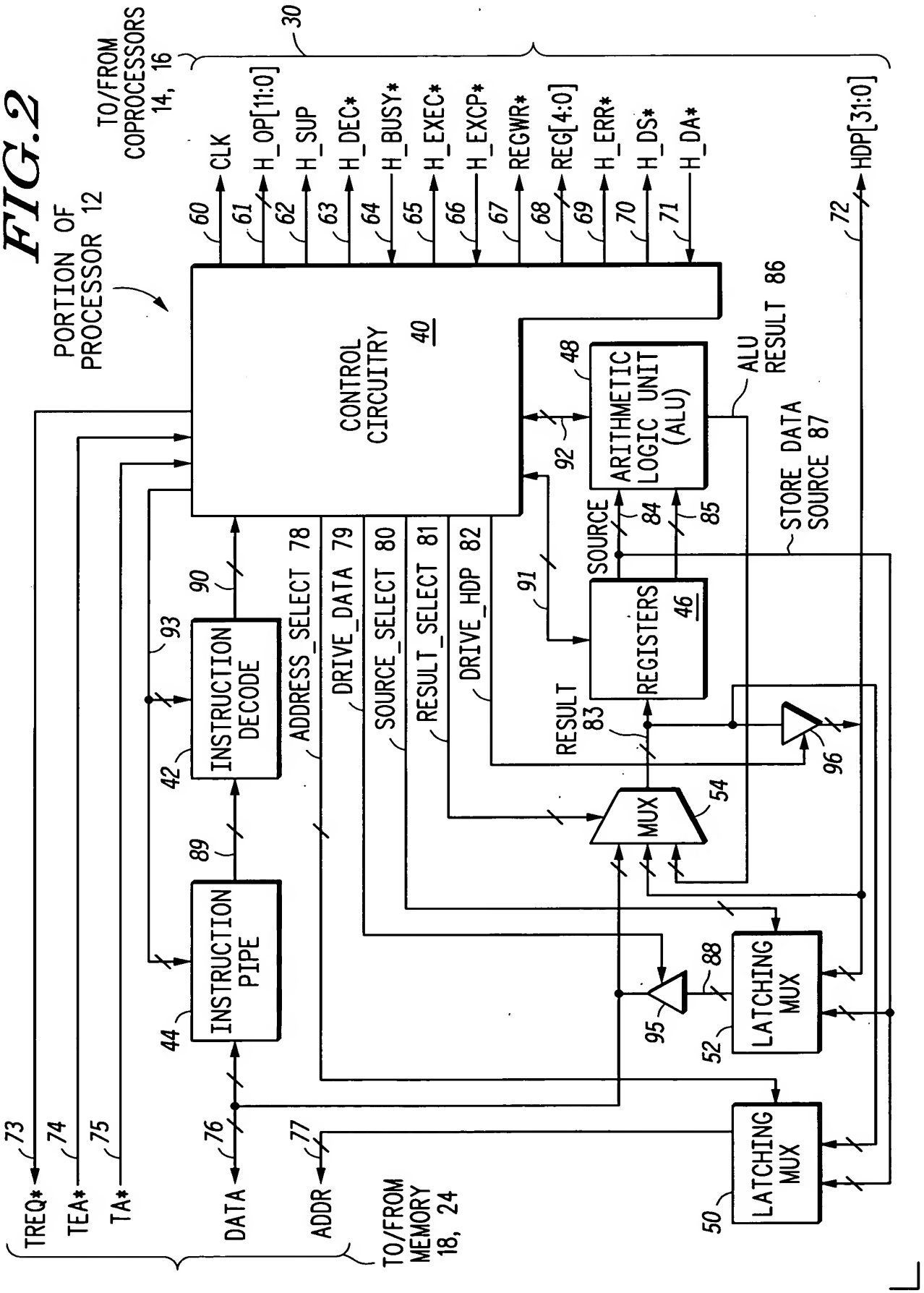


FIG.1

FIG.2


TO/FROM COPROCESSORS 14, 16
30
PORTION OF PROCESSOR 12
CONTROL CIRCUITRY
40
CLK
H_OP[11:0]
H_SUP
H_DEC*
H_BUSY*
H_EXEC*
H_EXCP*
REGWR*
REG[4:0]
H_ERR*
H_DS*
H_DA*
HDP[31:0]
ARITHMETIC LOGIC UNIT (ALU)
48
ALU RESULT 86
STORE DATA SOURCE 87
INSTRUCTION DECODE
42
INSTRUCTION PIPE
44
ADDRESS_SELECT 78
DRIVE_DATA 79
SOURCE_SELECT 80
RESULT_SELECT 81
DRIVE_HDP 82
SOURCE
REGISTERS
46
RESULT 83
MUX
54
LATCHING MUX
52
LATCHING MUX
50
TREQ*
TEA*
TA*
DATA
ADDR
TO/FROM MEMORY 18, 24

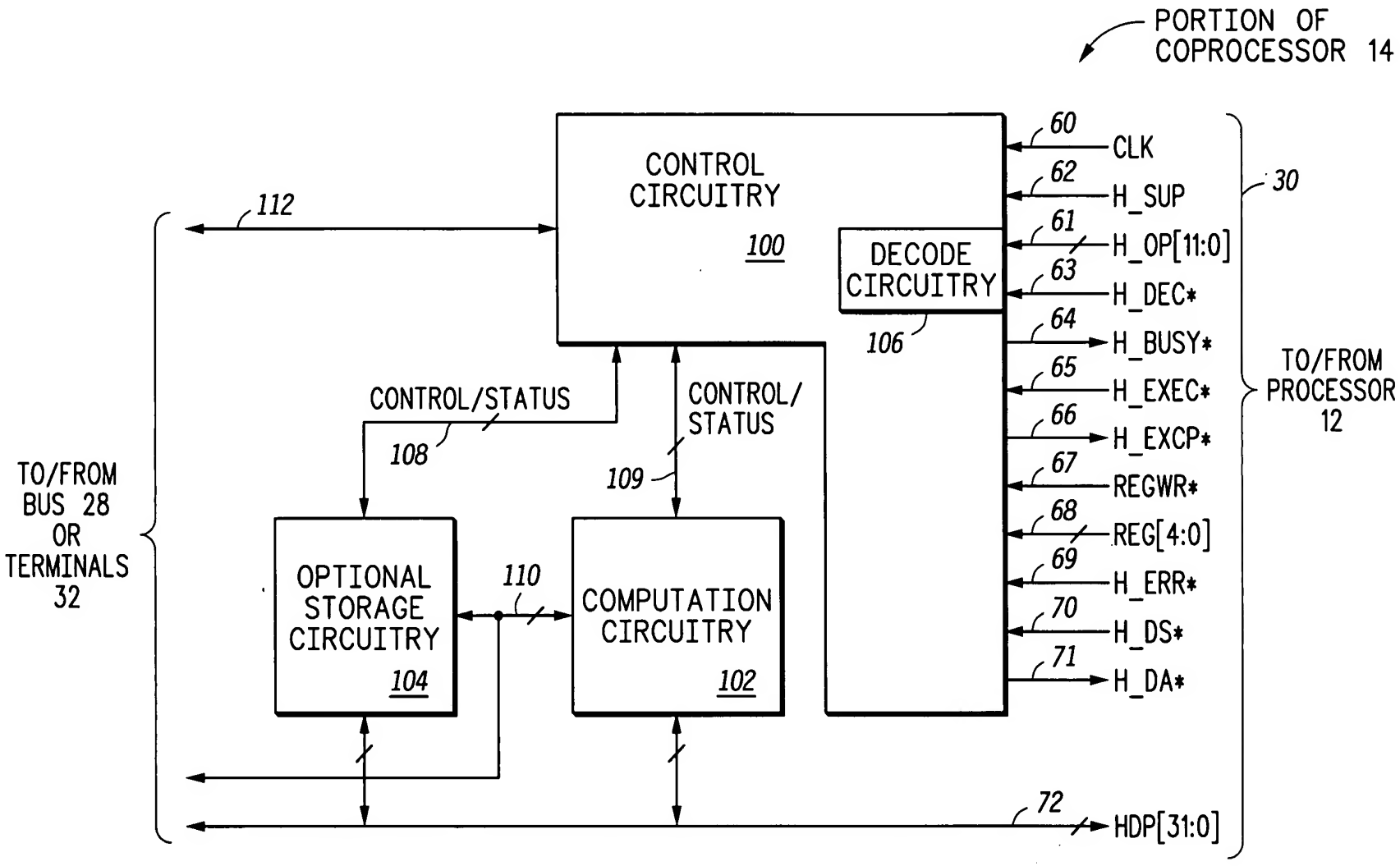

PORTION OF
COPROCESSOR 14
CONTROL
CIRCUITRY
100
DECODE
CIRCUITRY
106
112
60
CLK
62
H_SUP
61
H_OP[11:0]
63
H_DEC*
64
H_BUSY*
65
H_EXEC*
66
H_EXCP*
67
REGWR*
68
REG[4:0]
69
H_ERR*
70
H_DS*
71
H_DA*
30
TO/FROM
PROCESSOR
12
CONTROL/STATUS
108
CONTROL/
STATUS
109
TO/FROM
BUS 28
OR
TERMINALS
32
OPTIONAL
STORAGE
CIRCUITRY
104
110
COMPUTATION
CIRCUITRY
102
72
HDP[31:0]

FIG.3

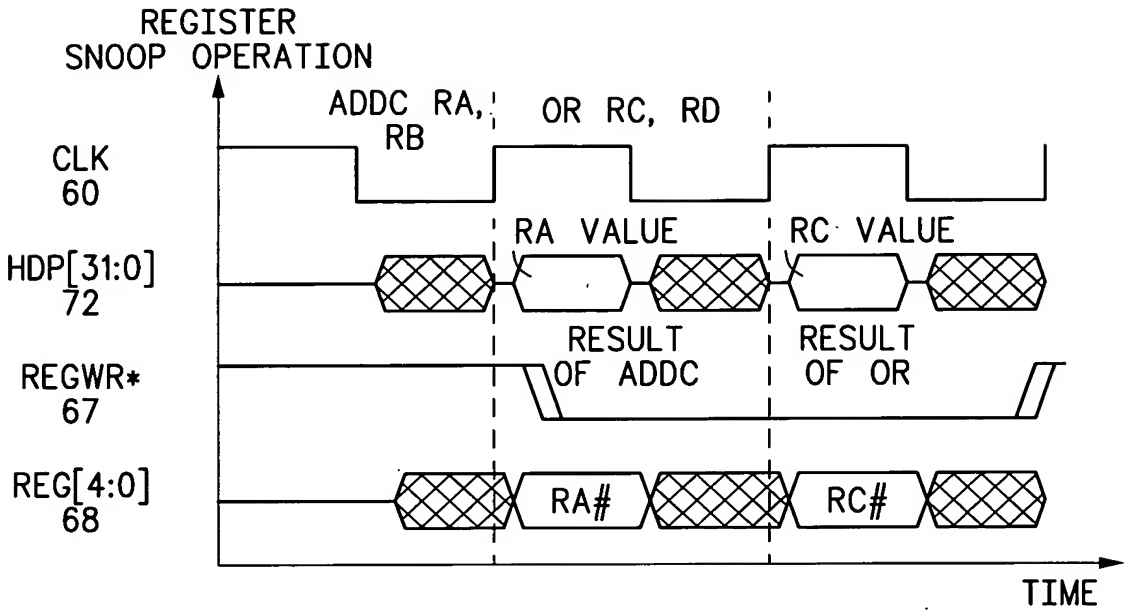

REGISTER
SNOOP OPERATION
ADDC RA, RB
OR RC, RD
CLK
60
RA VALUE
RC VALUE
HDP[31:0]
72
RESULT
OF ADDC
RESULT
OF OR
REGWR*
67
REG[4:0]
68
RA#
RC#
TIME

FIG.4

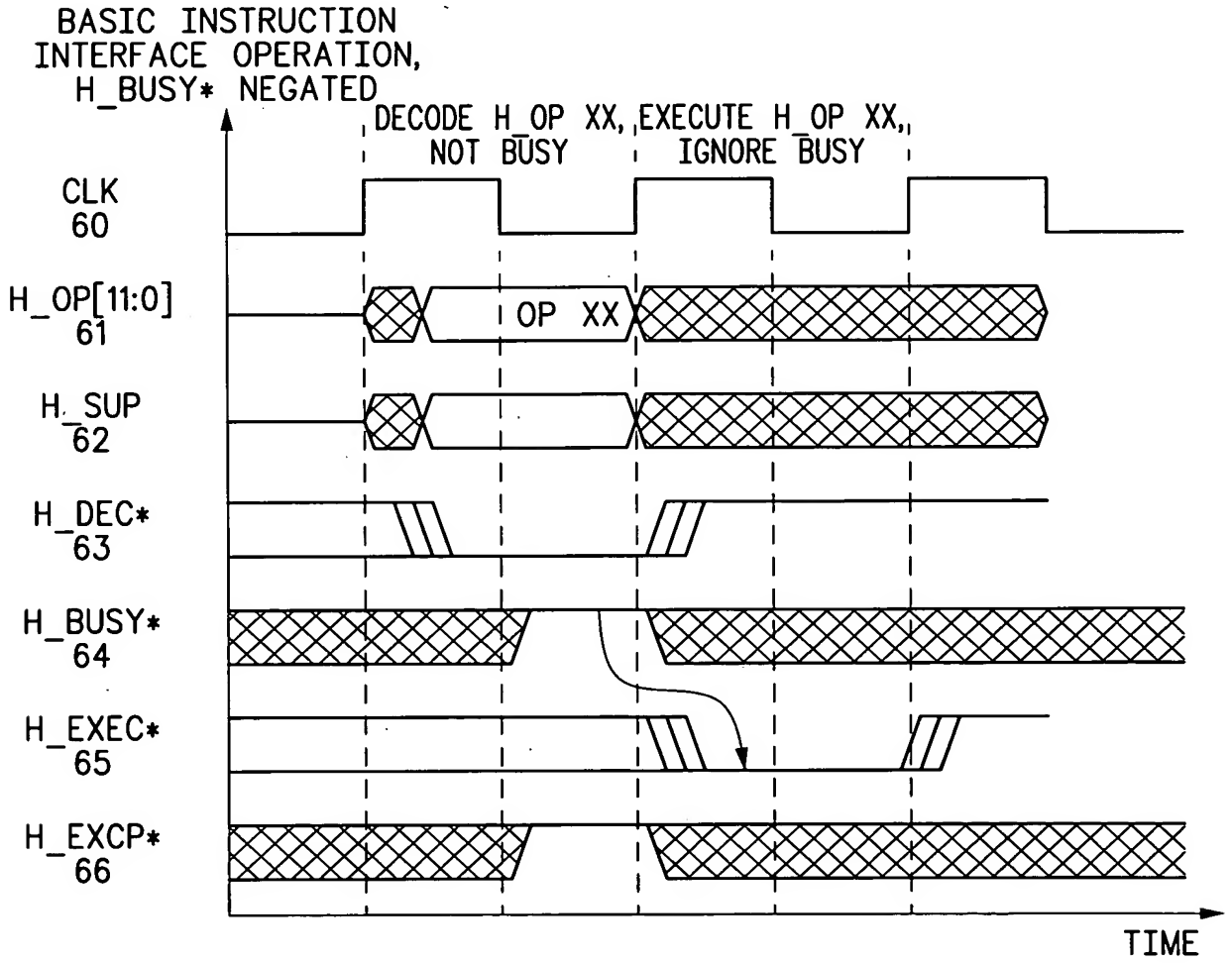

BASIC INSTRUCTION
INTERFACE OPERATION,
H_BUSY* NEGATED
DECODE H_OP XX,
NOT BUSY
EXECUTE H_OP XX,
IGNORE BUSY
CLK
60
H_OP[11:0]
61
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
TIME

FIG.5

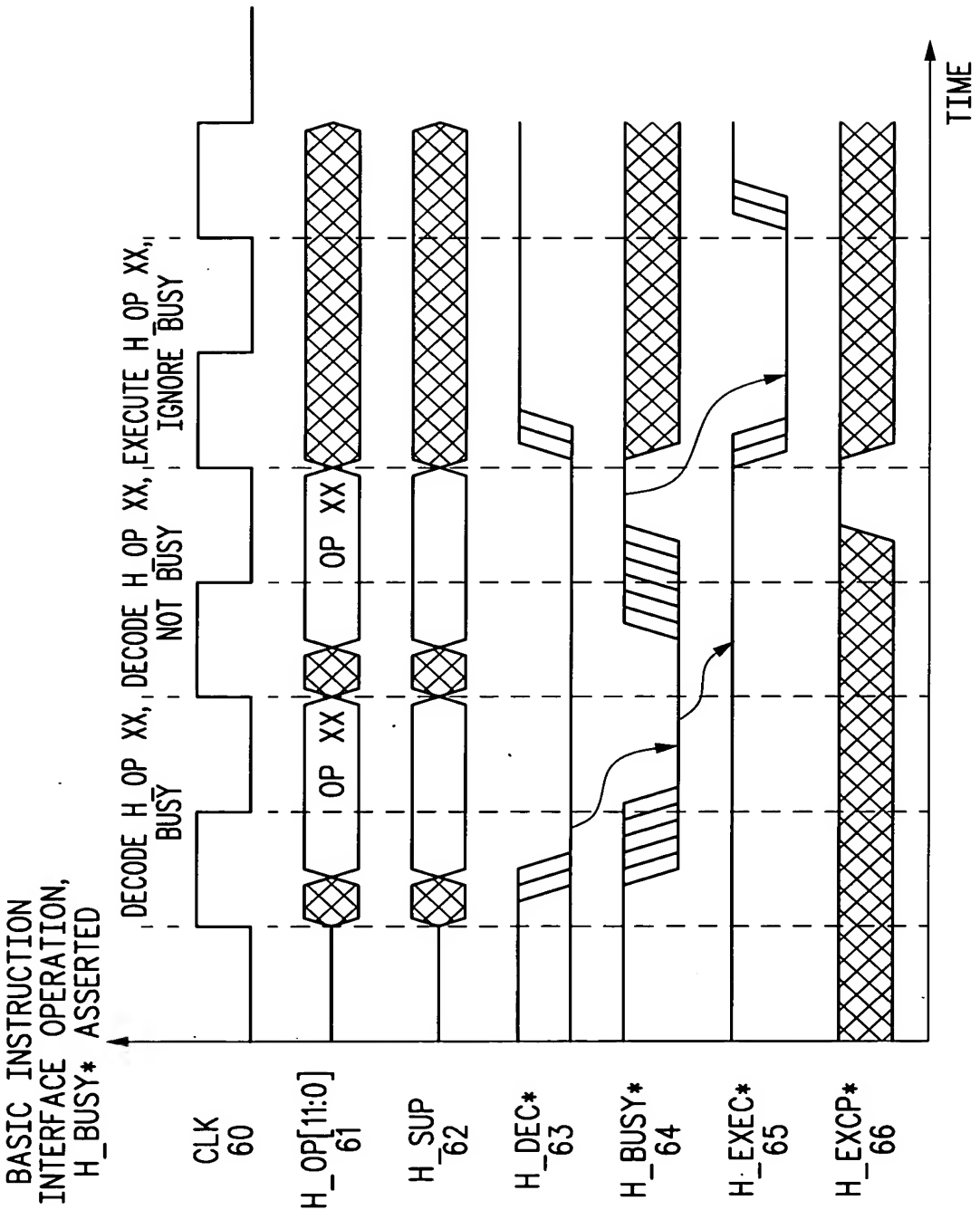

BASIC INSTRUCTION
INTERFACE OPERATION,
H_BUSY* ASSERTED
DECODE H_OP XX, EXECUTE H_OP XX,
DECODE H_OP XX,
BUSY
NOT BUSY
IGNORE BUSY
CLK
60
H_OP[11:0]
61
OP XX
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
TIME

FIG.6

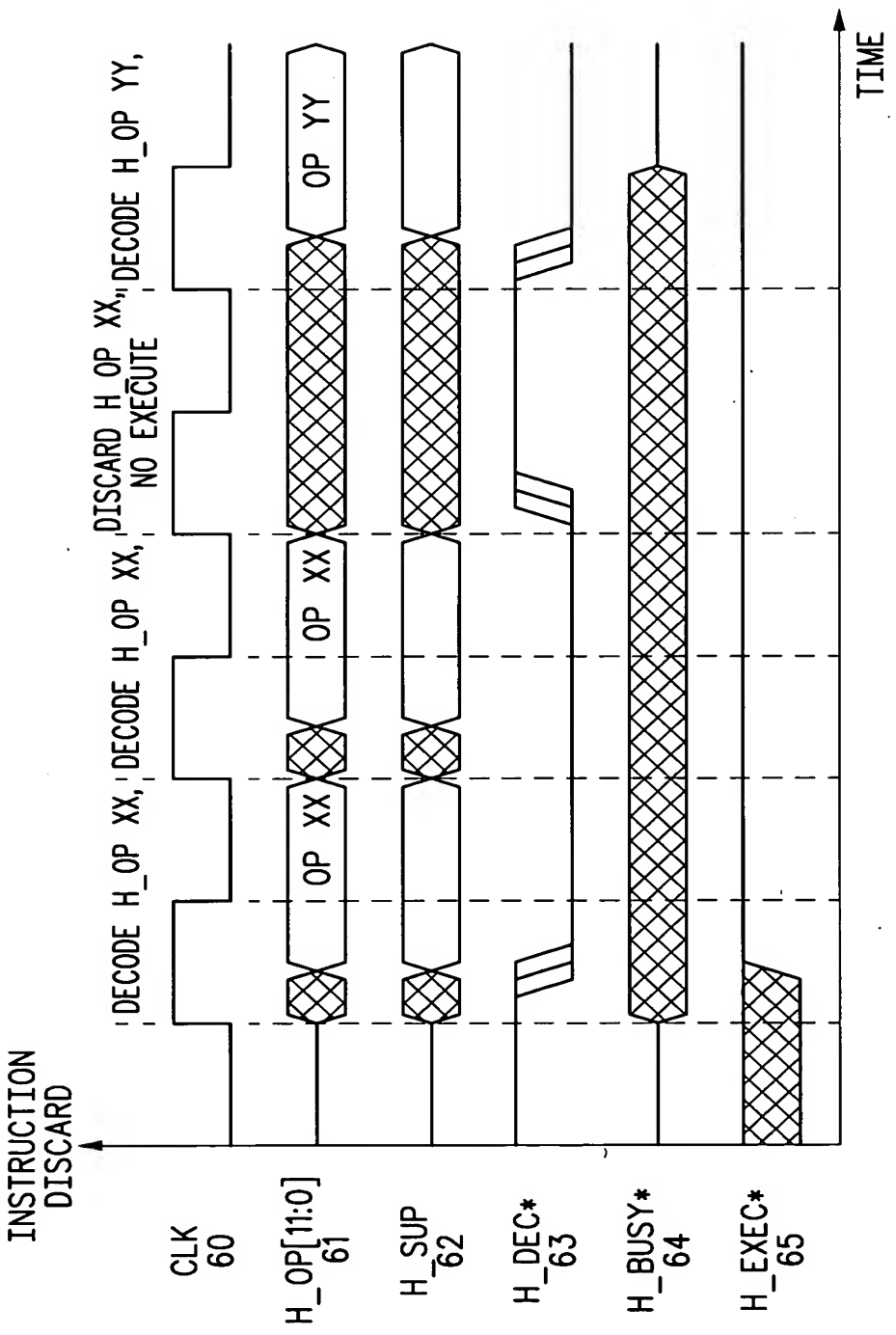

INSTRUCTION DISCARD
CLK 60
H_OP[11:0] 61
H_SUP 62
H_DEC* 63
H_BUSY* 64
H_EXEC* 65
DECODE H_OP XX,
DECODE H_OP XX,
DISCARD H_OP XX,
NO EXECUTE
DECODE H_OP YY,
OP XX
OP XX
OP YY
TIME

FIG. 7

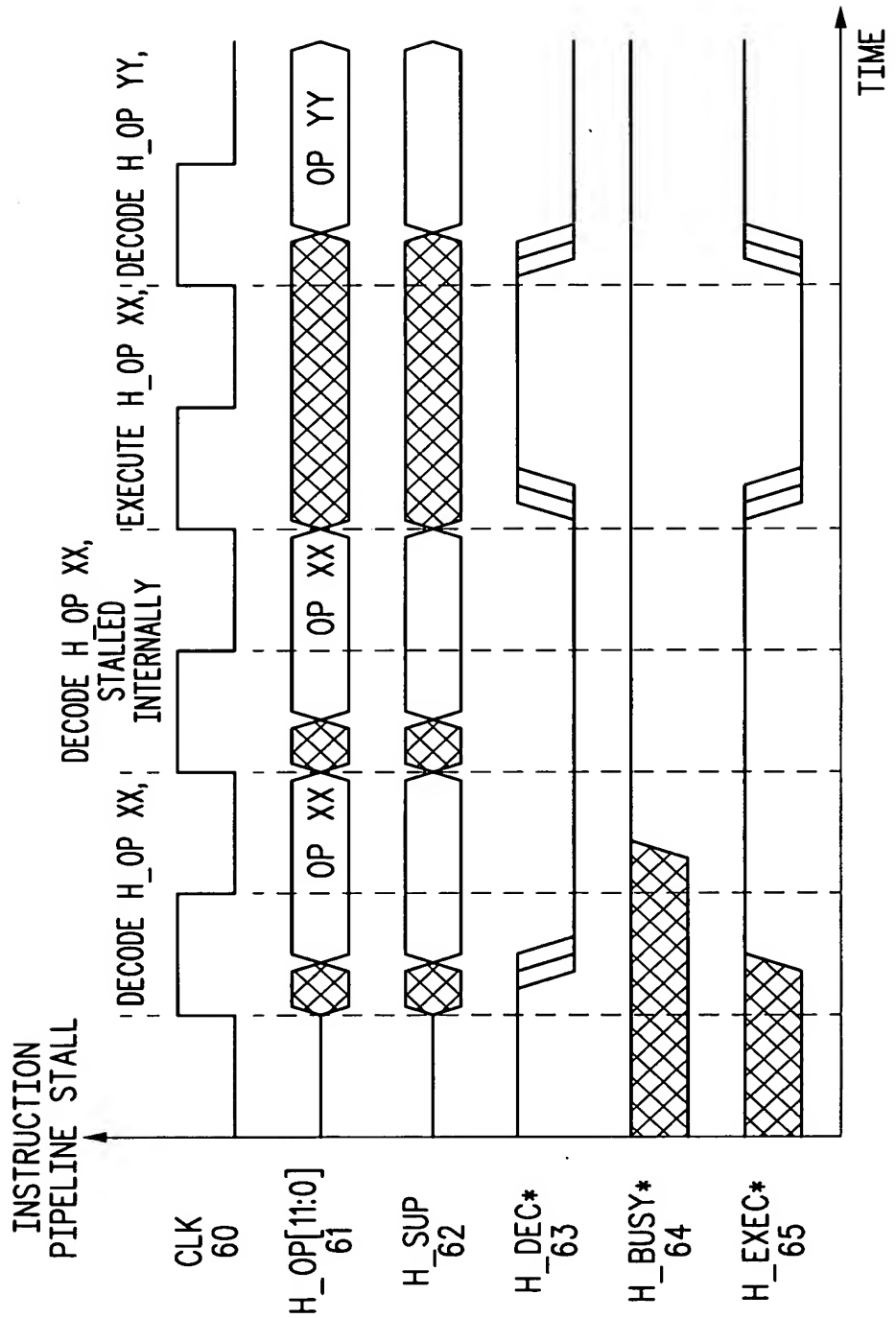

INSTRUCTION PIPELINE STALL
DECODE H_OP XX,
DECODE H_OP XX, STALLED INTERNALLY
EXECUTE H_OP XX, DECODE H_OP YY,
OP XX
OP XX
OP YY
CLK 60
H_OP[11:0] 61
H_SUP 62
H_DEC* 63
H_BUSY* 64
H_EXEC* 65
TIME

FIG.8

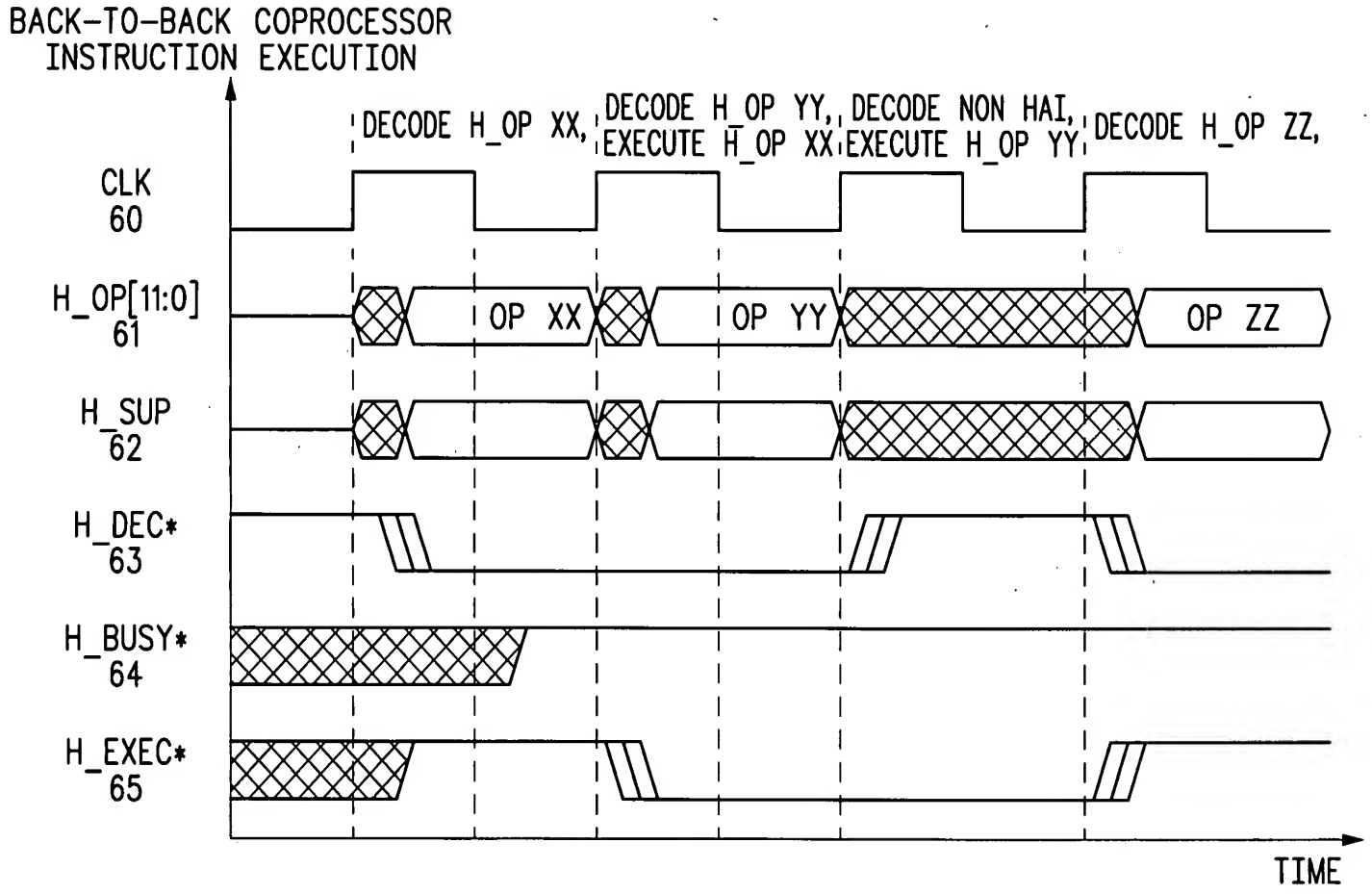

BACK-TO-BACK COPROCESSOR
INSTRUCTION EXECUTION
DECODE H_OP XX,
DECODE H_OP YY,
EXECUTE H_OP XX
DECODE NON HAI,
EXECUTE H_OP YY
DECODE H_OP ZZ,
CLK
60
H_OP[11:0]
61
OP XX
OP YY
OP ZZ
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
TIME

FIG.9

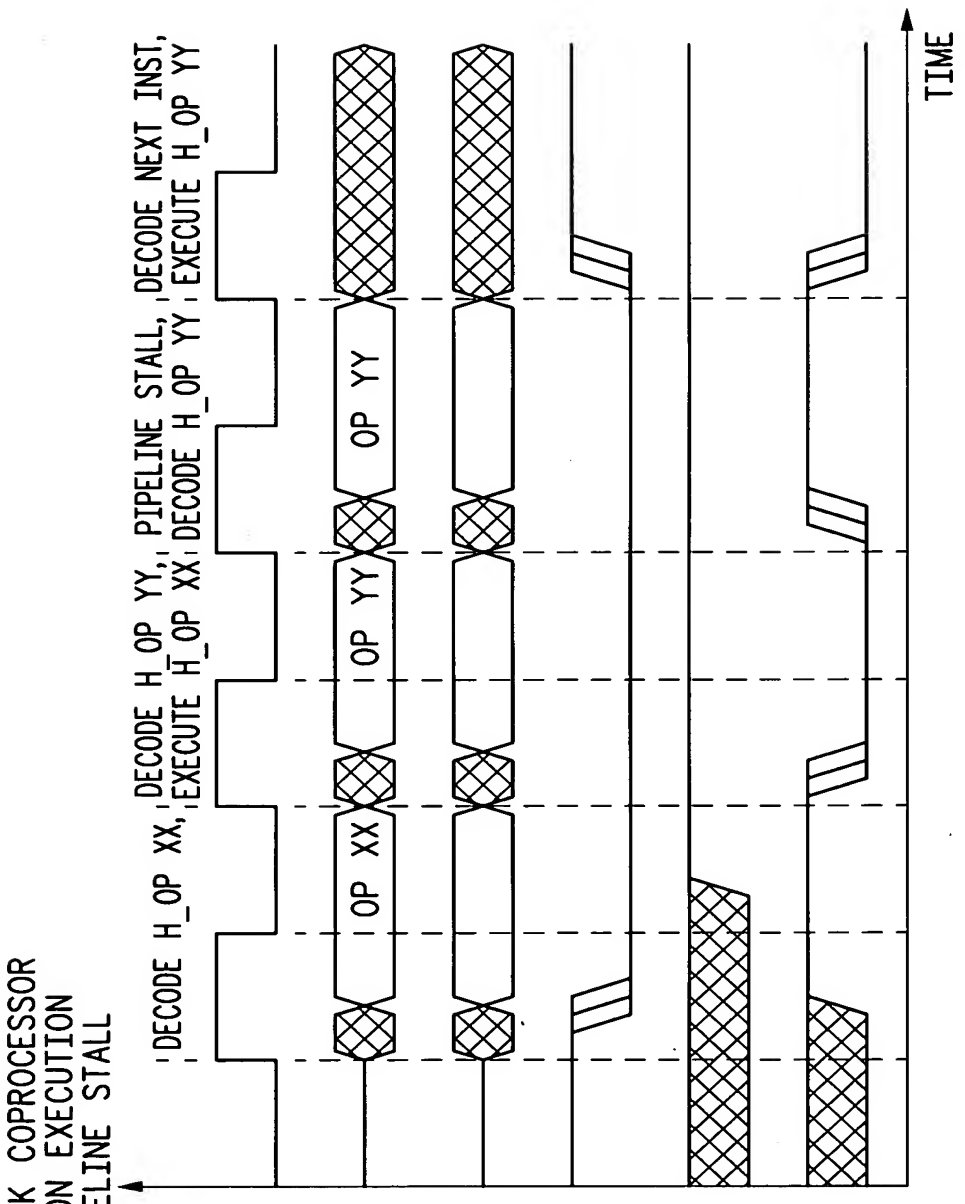

BACK-TO-BACK COPROCESSOR
INSTRUCTION EXECUTION
WITH PIPELINE STALL
DECODE H_OP XX, DECODE H_OP YY, PIPELINE STALL, DECODE NEXT INST,
EXECUTE H_OP XX, DECODE H_OP YY, EXECUTE H_OP YY
CLK
60
H_OP[11:0]
61
OP XX
OP YY
OP YY
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
TIME

FIG.10

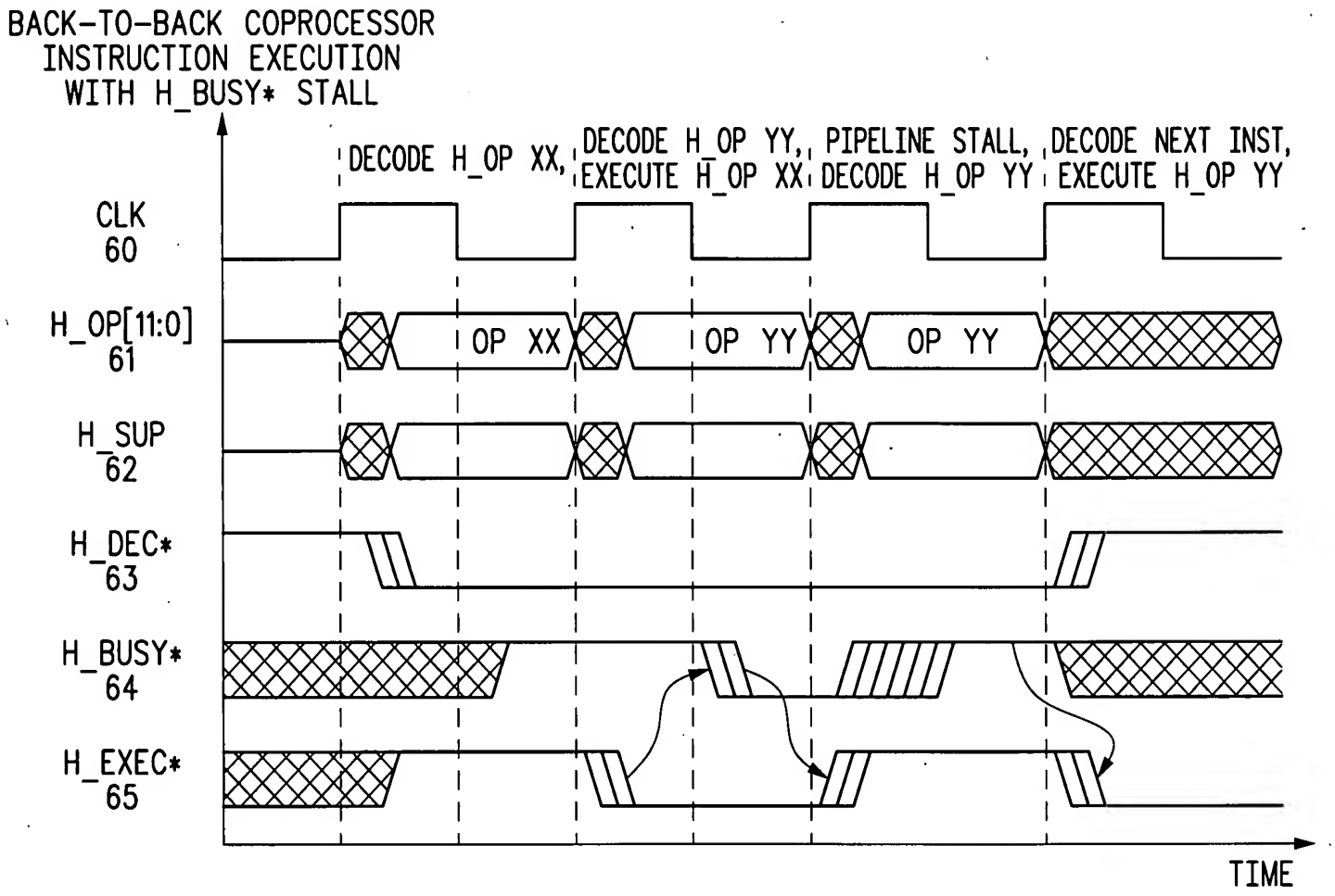

BACK-TO-BACK COPROCESSOR
INSTRUCTION EXECUTION
WITH H_BUSY* STALL
DECODE H_OP XX,
DECODE H_OP YY,
EXECUTE H_OP XX
PIPELINE STALL,
DECODE H_OP YY
DECODE NEXT INST,
EXECUTE H_OP YY
CLK
60
H_OP[11:0]
61
OP XX
OP YY
OP YY
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
TIME

FIG.11

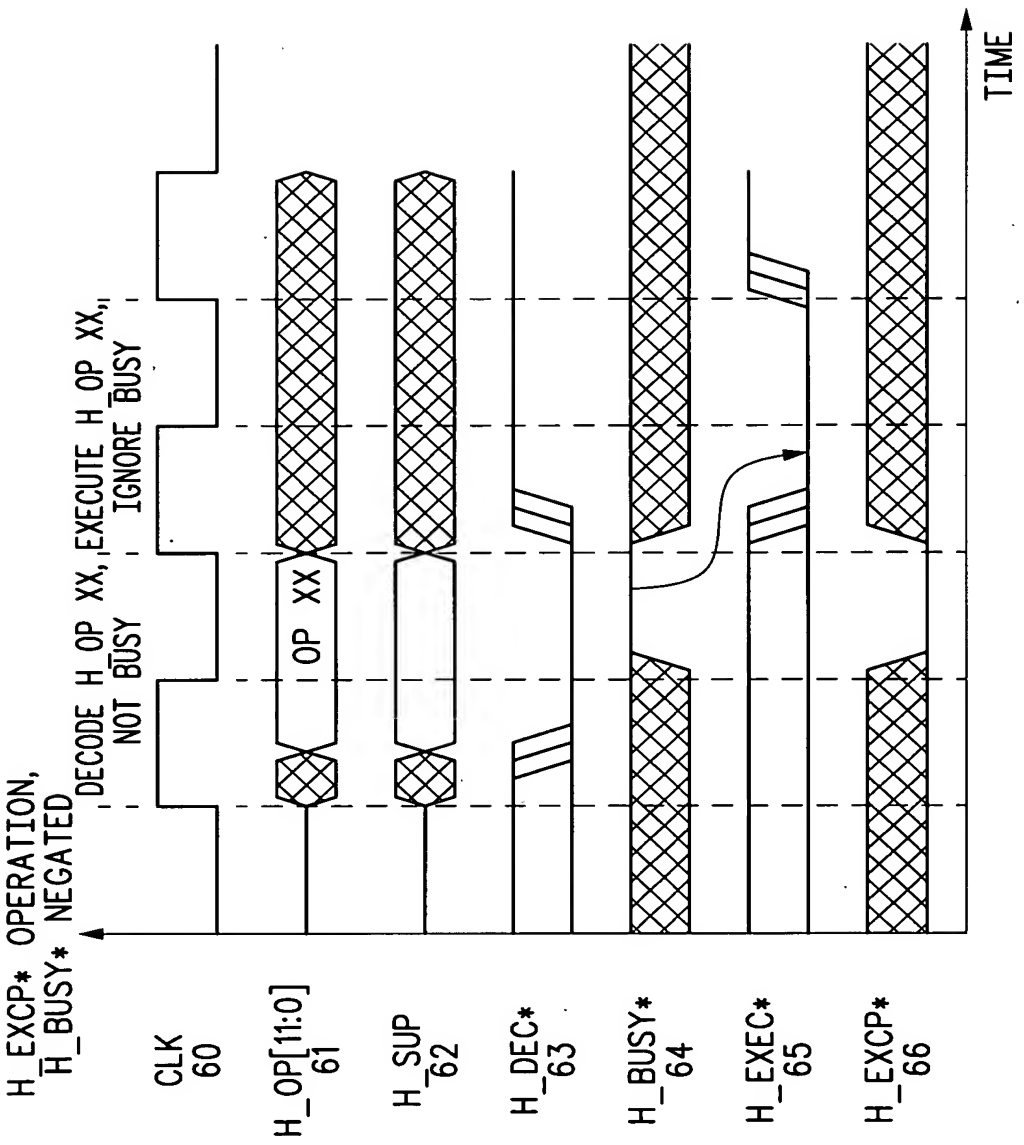

H_EXCP* OPERATION,
H_BUSY* NEGATED
DECODE H OP XX, EXECUTE H OP XX,
NOT BUSY IGNORE BUSY
CLK
60
H_OP[11:0]
61
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
TIME

FIG.12

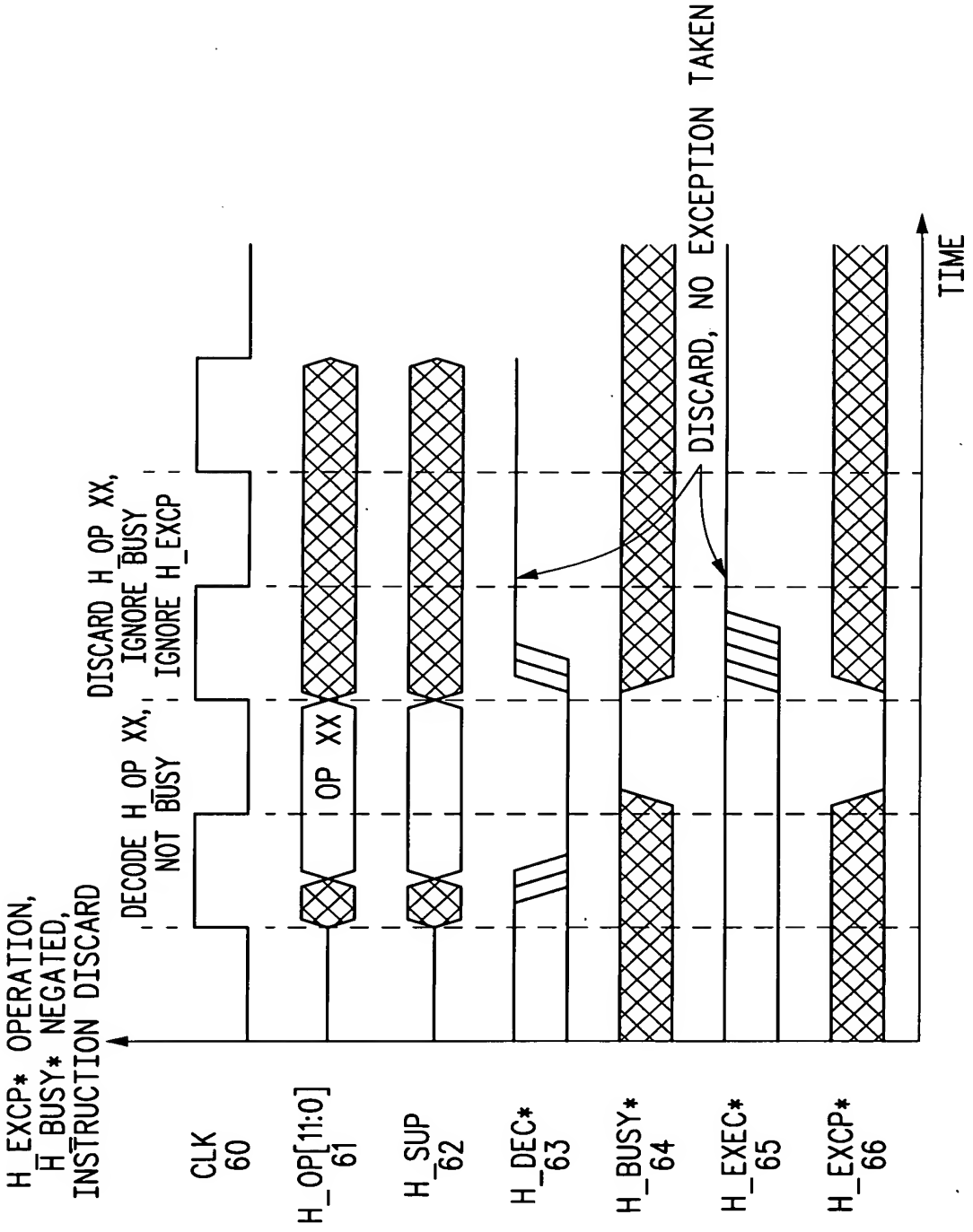

H_EXCP* OPERATION,
H_BUSY* NEGATED,
INSTRUCTION DISCARD
DECODE H_OP XX,
NOT BUSY
DISCARD H_OP XX,
IGNORE BUSY
IGNORE H_EXCP
DISCARD, NO EXCEPTION TAKEN
TIME
CLK
60
H_OP[11:0]
61
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66

FIG.13

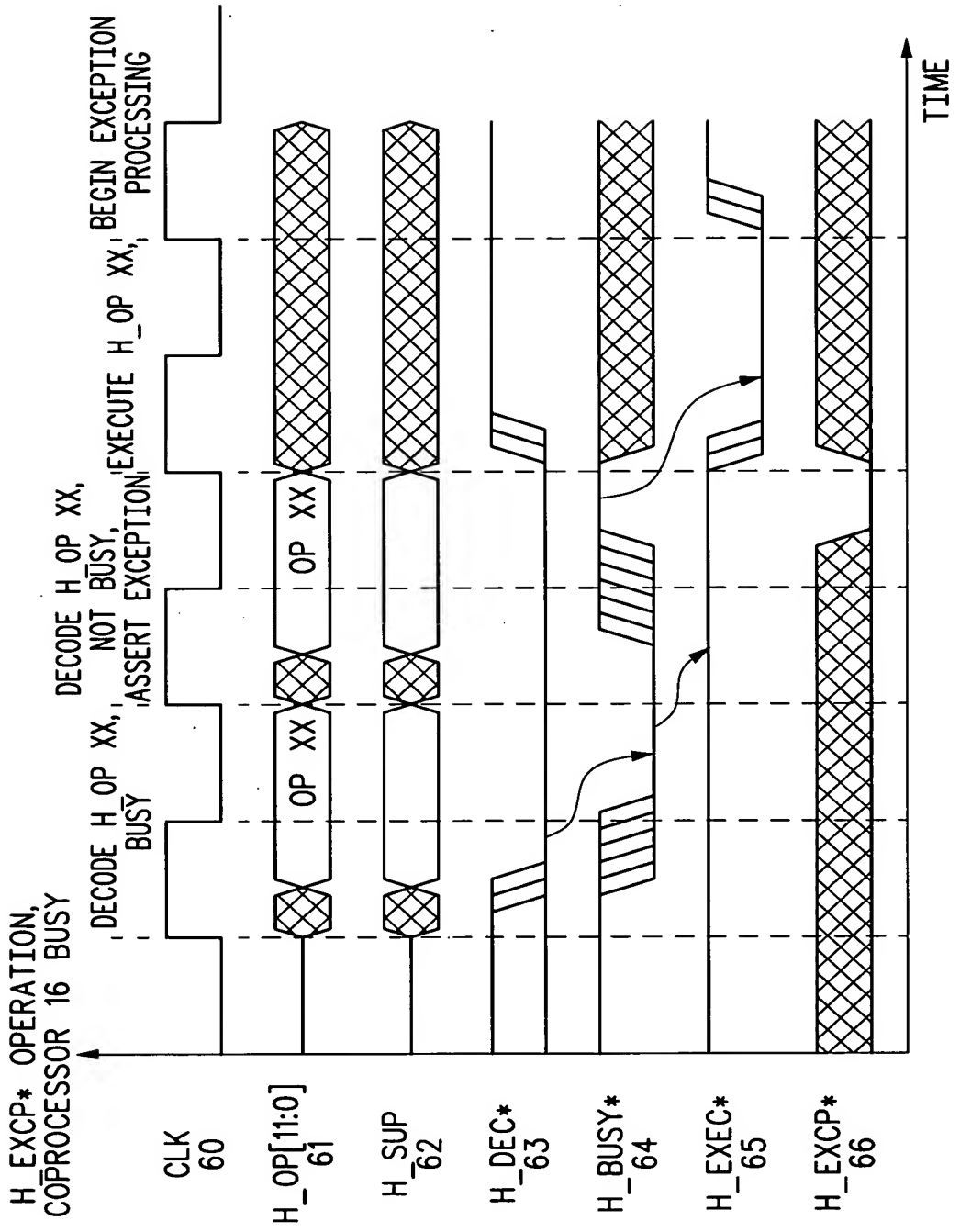

H_EXCP* OPERATION,
COPROCESSOR 16 BUSY
DECODE H_OP XX,
BUSY
DECODE H_OP XX,
NOT BUSY,
ASSERT EXCEPTION
EXECUTE H_OP XX,
BEGIN EXCEPTION
PROCESSING
CLK
60
H_OP[11:0]
61
OP XX
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
TIME

FIG.14

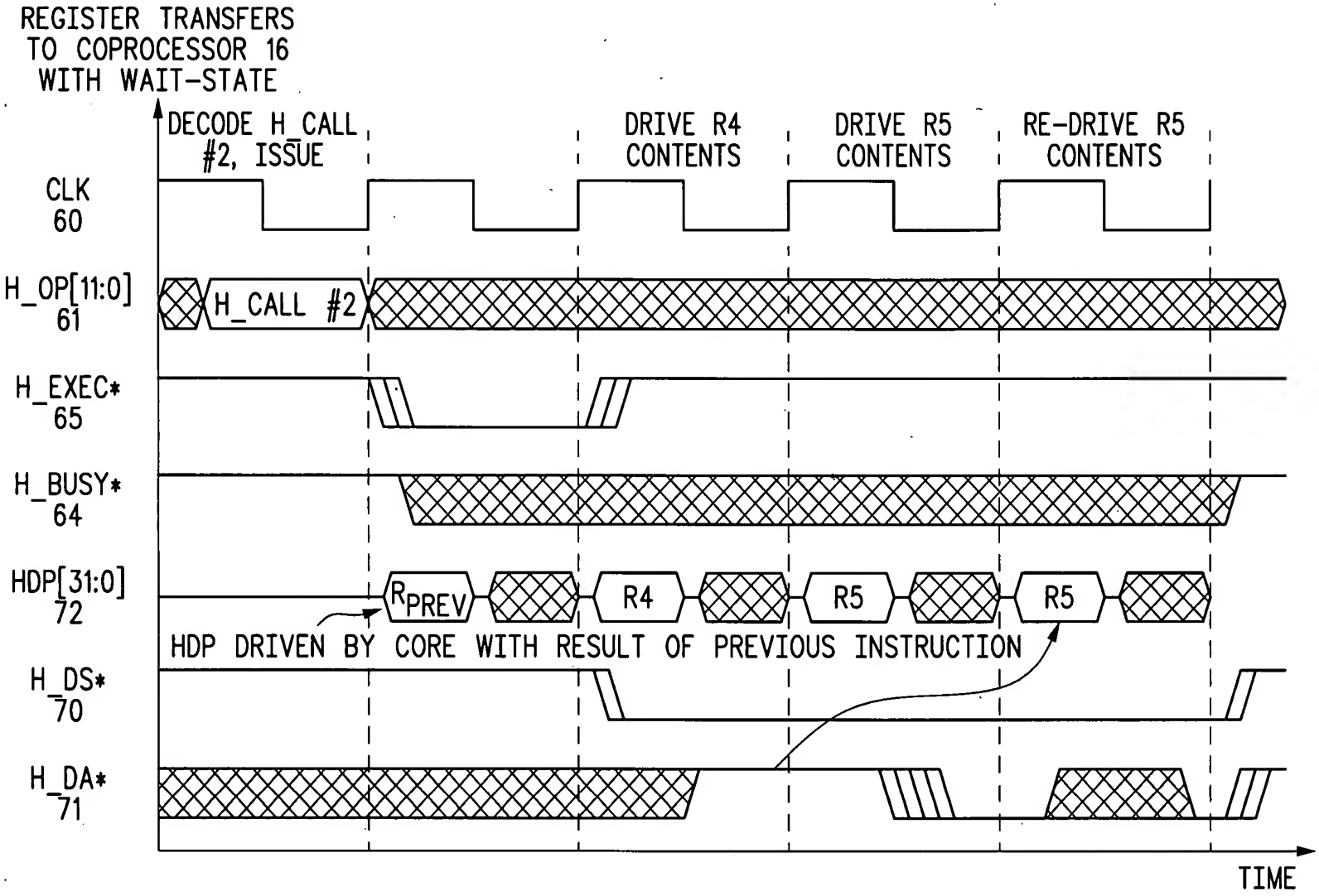

REGISTER TRANSFERS
TO COPROCESSOR 16
WITH WAIT-STATE
DECODE H_CALL
#2, ISSUE
DRIVE R4
CONTENTS
DRIVE R5
CONTENTS
RE-DRIVE R5
CONTENTS
CLK
60
H_OP[11:0]
61
H_CALL #2
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
R4
R5
R5
HDP DRIVEN BY CORE WITH RESULT OF PREVIOUS INSTRUCTION
H_DS*
70
H_DA*
71
TIME

FIG.15

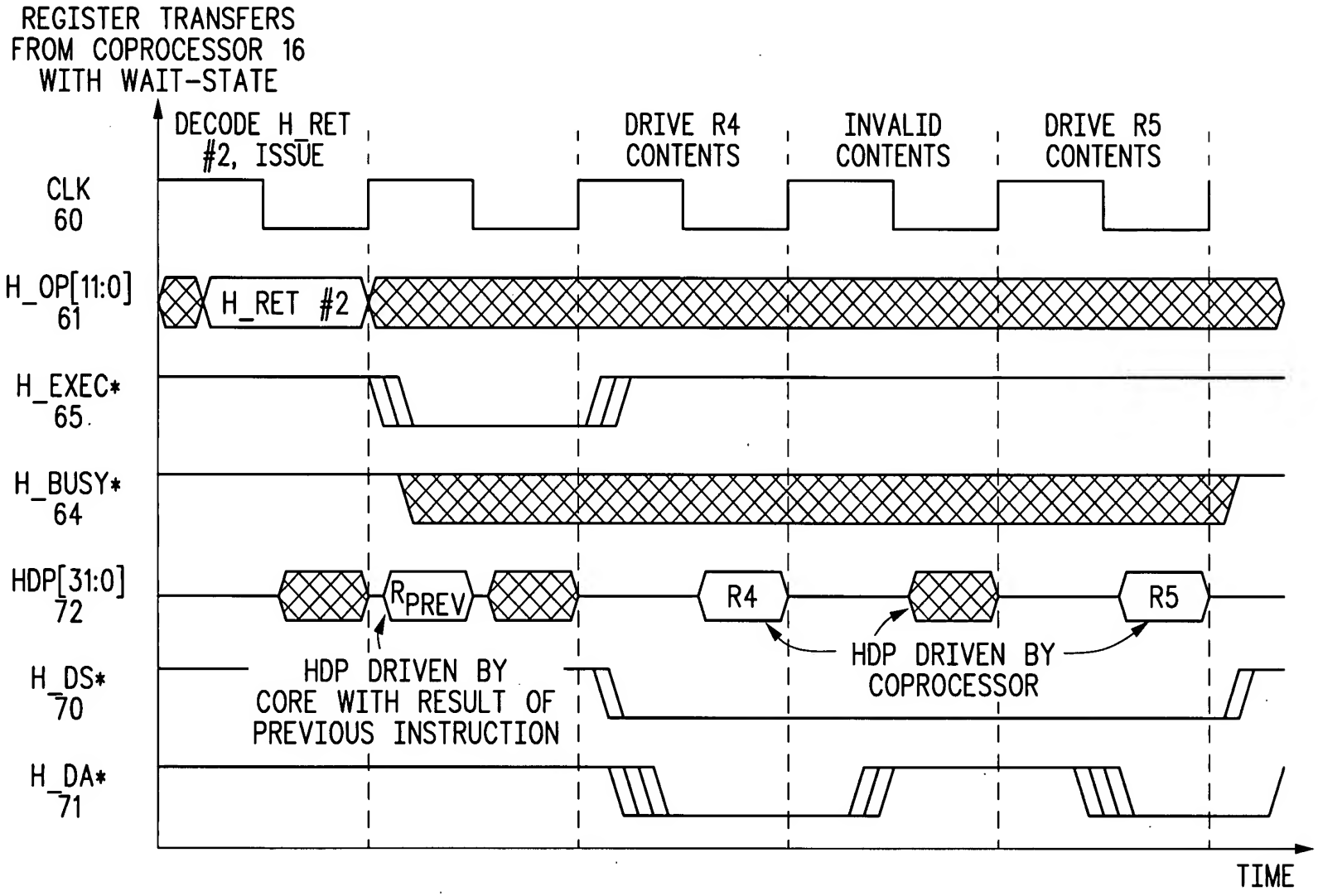

REGISTER TRANSFERS
FROM COPROCESSOR 16
WITH WAIT-STATE
DECODE H_RET
#2, ISSUE
DRIVE R4
CONTENTS
INVALID
CONTENTS
DRIVE R5
CONTENTS
CLK
60
H_OP[11:0]
61
H_RET #2
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
R4
R5
HDP DRIVEN BY
COPROCESSOR
H_DS*
70
HDP DRIVEN BY
CORE WITH RESULT OF
PREVIOUS INSTRUCTION
H_DA*
71
TIME

FIG.16

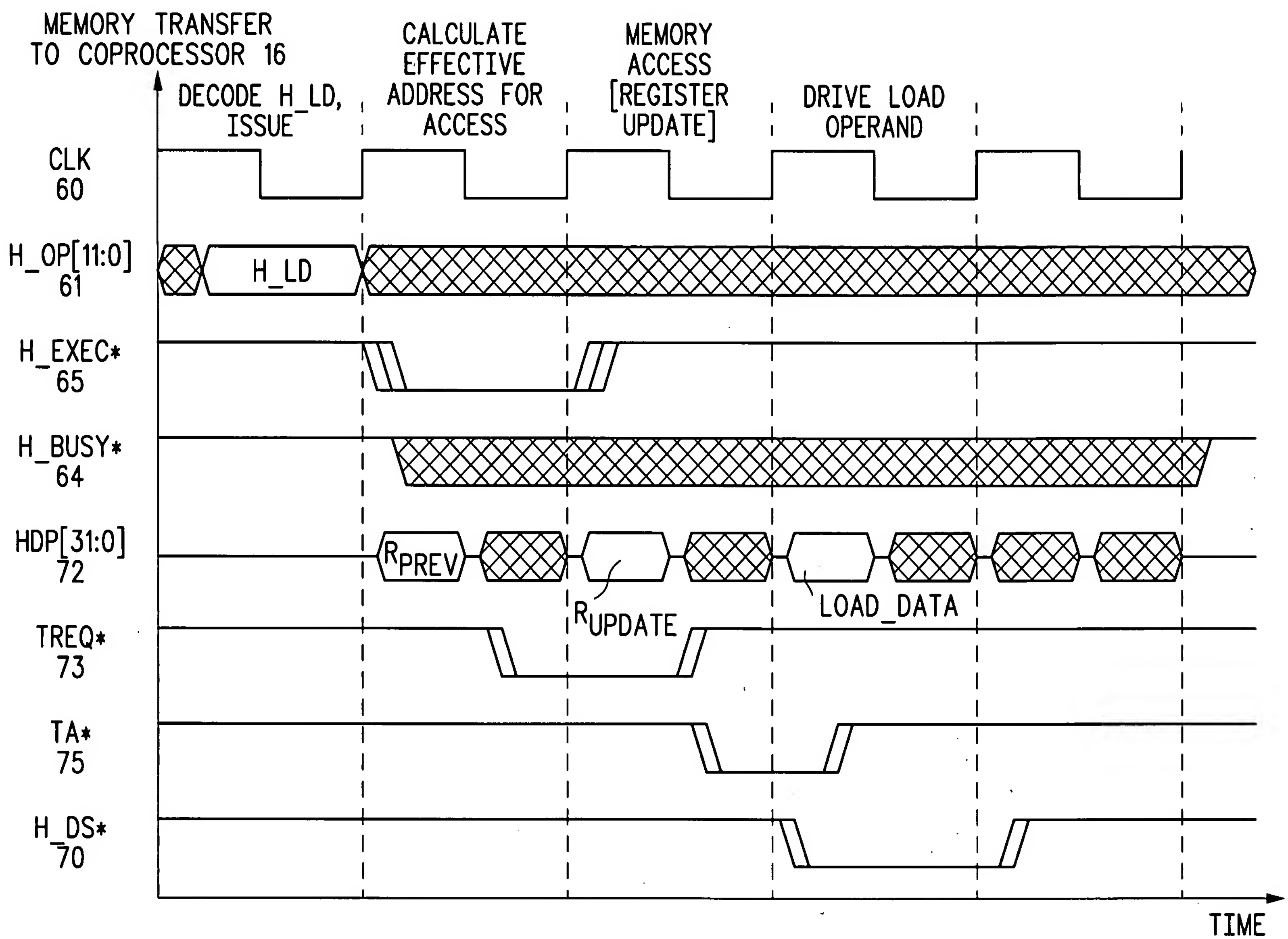

MEMORY TRANSFER TO COPROCESSOR 16
DECODE H_LD, ISSUE
CALCULATE EFFECTIVE ADDRESS FOR ACCESS
MEMORY ACCESS [REGISTER UPDATE]
DRIVE LOAD OPERAND
CLK 60
H_OP[11:0] 61
H_LD
H_EXEC* 65
H_BUSY* 64
HDP[31:0] 72
R PREV
R UPDATE
LOAD_DATA
TREQ* 73
TA* 75
H_DS* 70
TIME

FIG.17

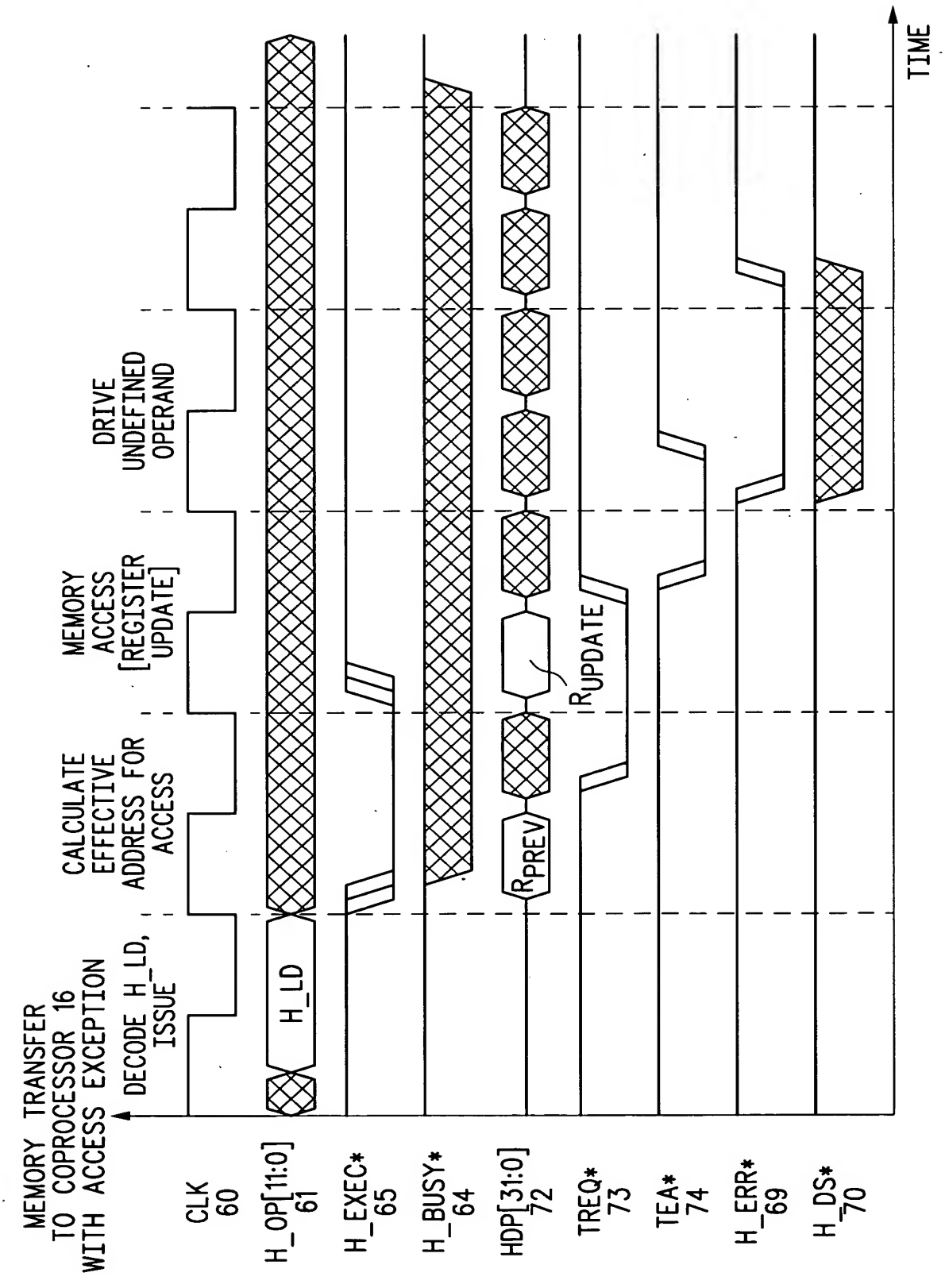

MEMORY TRANSFER
TO COPROCESSOR 16
WITH ACCESS EXCEPTION
DECODE H_LD,
ISSUE
CALCULATE
EFFECTIVE
ADDRESS FOR
ACCESS
MEMORY
ACCESS
[REGISTER
UPDATE]
DRIVE
UNDEFINED
OPERAND
TIME
CLK
60
H_OP[11:0]
61
H_LD
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
RUPDATE
TREQ*
73
TEA*
74
H_ERR*
69
H_DS*
70

FIG.18

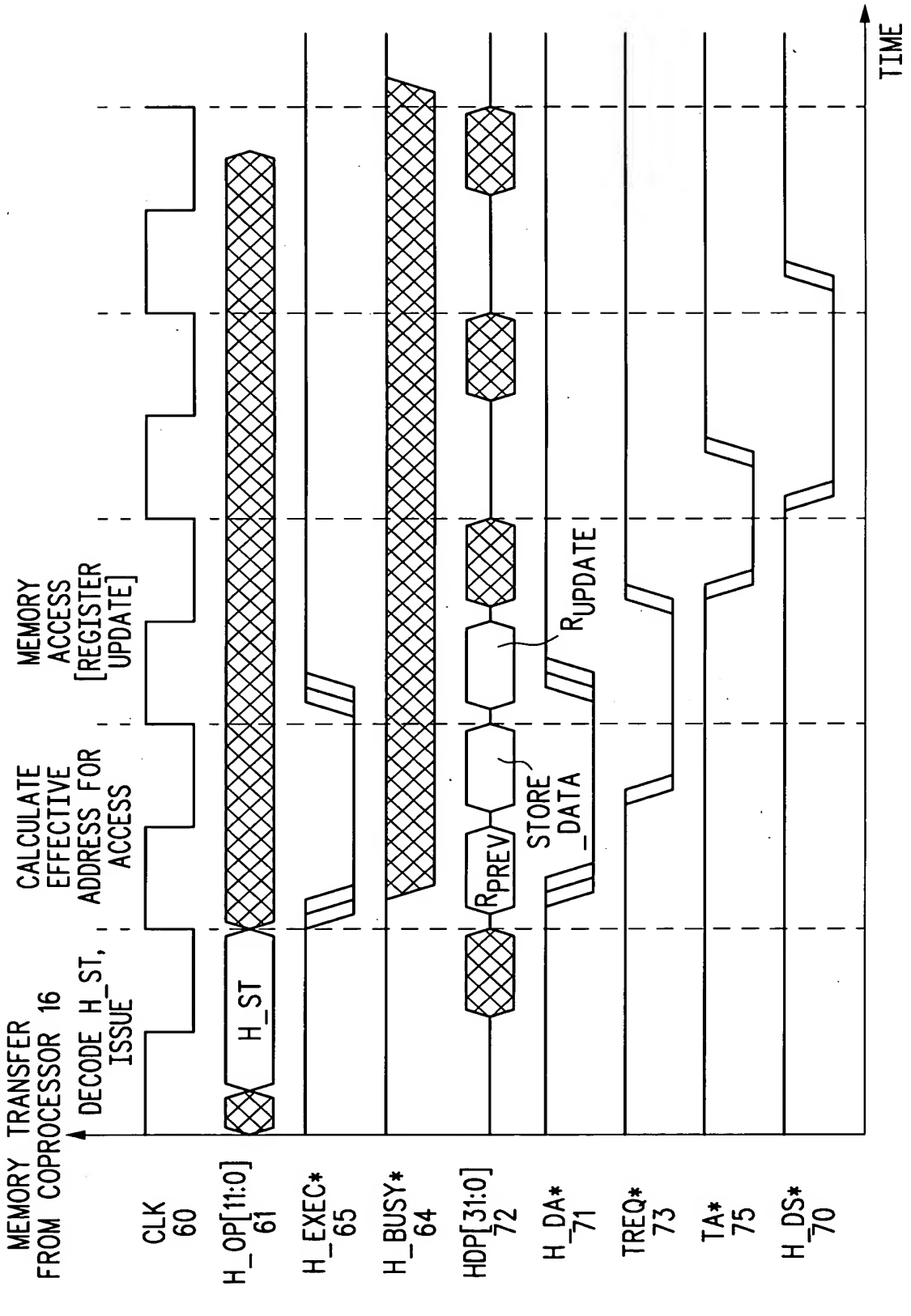

MEMORY TRANSFER FROM COPROCESSOR 16
DECODE H_ST, ISSUE
CALCULATE EFFECTIVE ADDRESS FOR ACCESS
MEMORY ACCESS [REGISTER UPDATE]
H_ST
RPREV
STORE _DATA
RUPDATE
CLK 60
H_OP[11:0] 61
H_EXEC* 65
H_BUSY* 64
HDP[31:0] 72
H_DA* 71
TREQ* 73
TA* 75
H_DS* 70
TIME

FIG.19

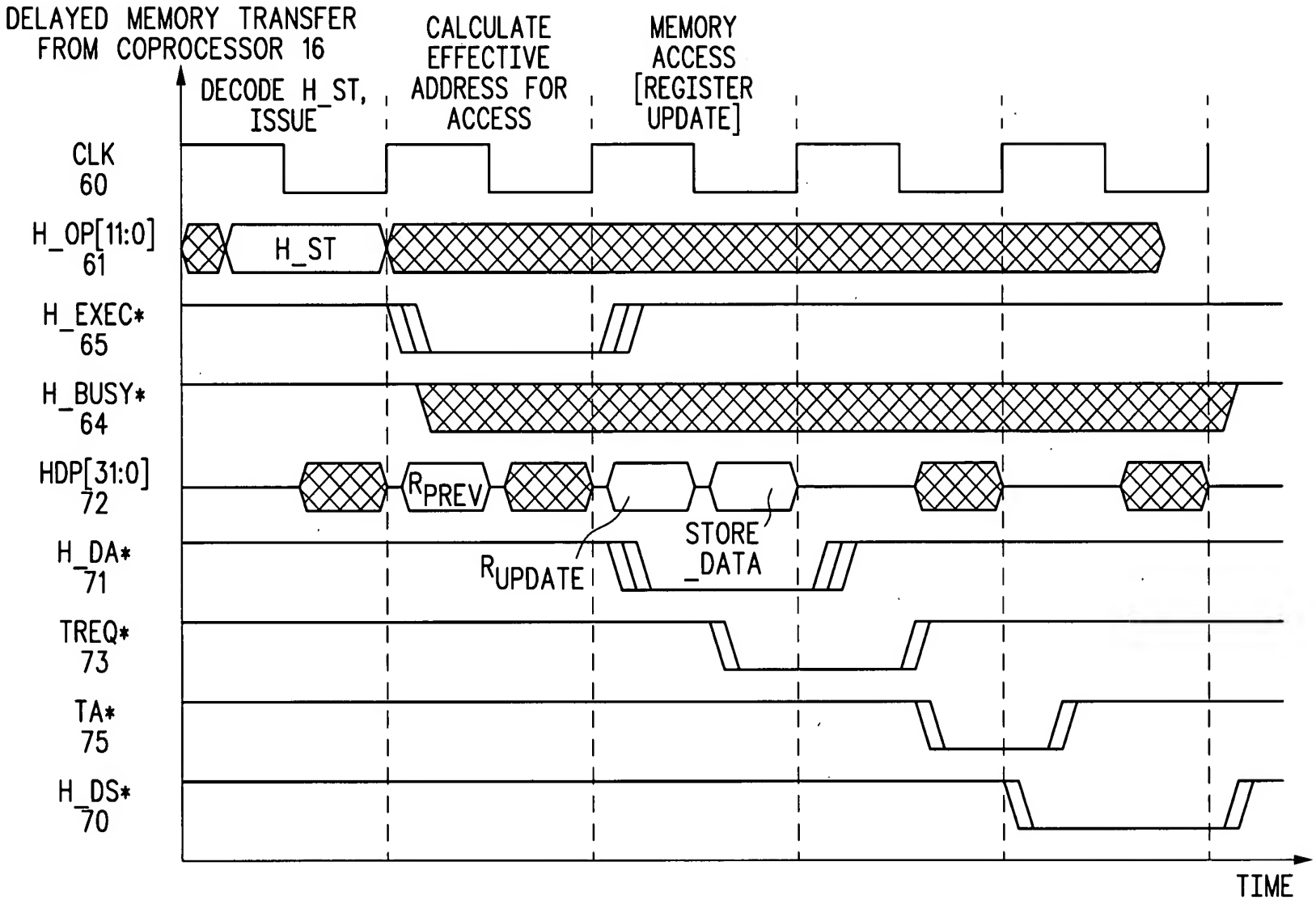

DELAYED MEMORY TRANSFER FROM COPROCESSOR 16
DECODE H_ST, ISSUE
CALCULATE EFFECTIVE ADDRESS FOR ACCESS
MEMORY ACCESS [REGISTER UPDATE]
CLK 60
H_OP[11:0] 61
H_ST
H_EXEC* 65
H_BUSY* 64
HDP[31:0] 72
R_PREV
R_UPDATE
STORE _DATA
H_DA* 71
TREQ* 73
TA* 75
H_DS* 70
TIME

FIG.20

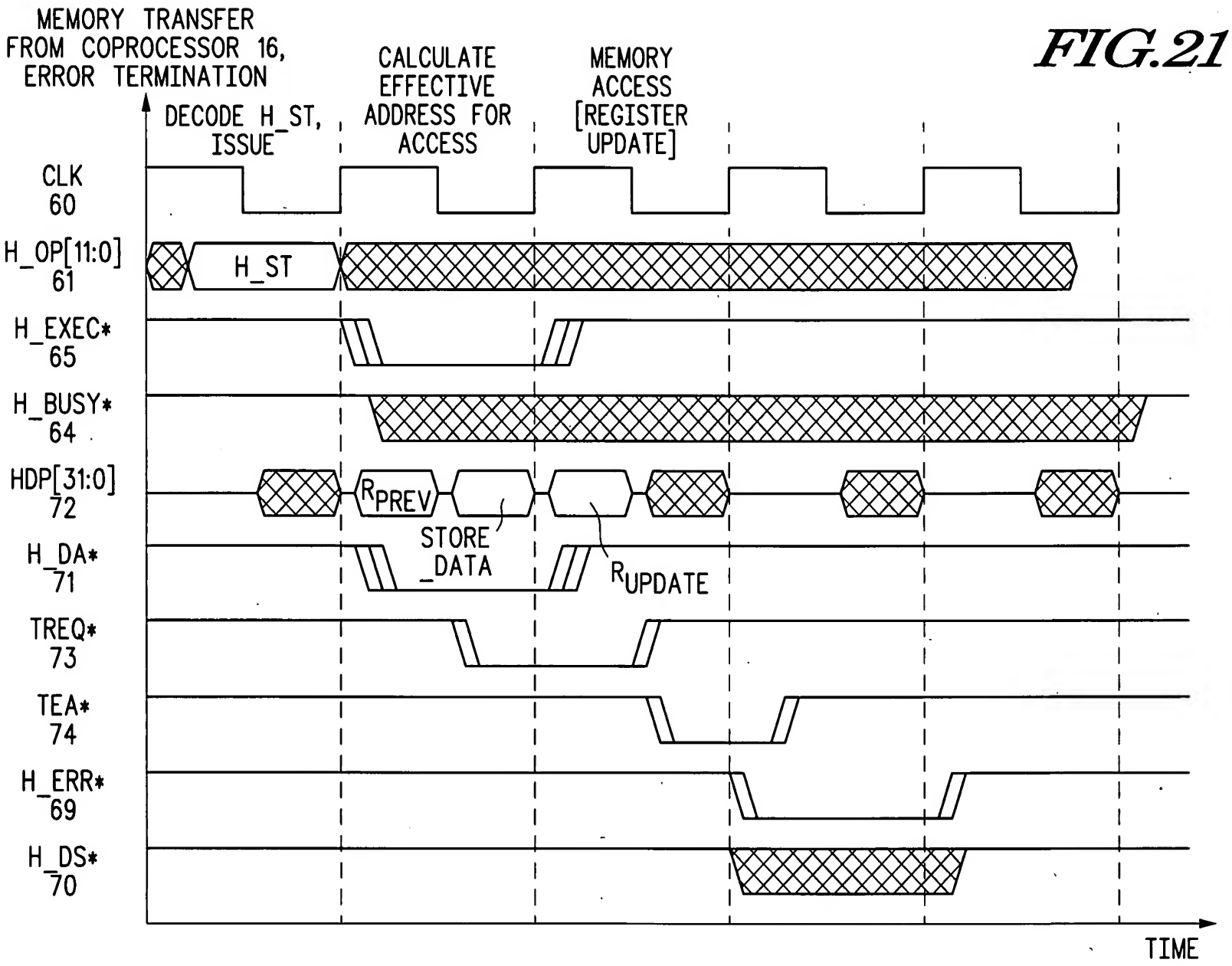


FIG.21

H_CALL HARDWARE ACCELERATOR (COPROCESSOR) CALL PRIMITIVE

OPERATION: PASS PARAMETERS TO HARDWARE ACCELERATOR

ASSEMBLER SYNTAX: H_CALL #UU, R4-RLAST, #CODE

DESCRIPTION: H_CALL PASSES A SET OF REGISTER-BASED PARAMETERS AND A CODE TO HARDWARE BLOCK (COPROCESSOR) #UU

CONDITION-CODE: UNAFFECTED

INSTRUCTION FORMAT:

15	14	13	12	11 10	9	8	7	6 5 4	3 2 1 0
0	1	0	0	UU	0	1	1	CNT	CODE

INSTRUCTION FIELDS:

UU FIELD-SPECIFIES HARDWARE BLOCK (COPROCESSOR)
- 00 - BLOCK 0
- 01 - BLOCK 1
- 10 - BLOCK 2
- 11 - BLOCK 3

CNT FIELD-SPECIFIES NUMBER OF REGISTERS TO PASS, BEGINNING WITH R4
- 000 - RESERVED, DO NOT USE
- 001 - PASS R4
- :
- 111 - PASS R4-R10

FIG.22

H_RET	HARDWARE ACCELERATOR (COPROCESSOR) RETURN PRIMITIVE
OPERATION:	PASS PARAMETERS FROM HARDWARE ACCELERATOR
ASSEMBLER SYNTAX:	H_RET #UU, R4–RLAST, #CODE
DESCRIPTION:	H_RET PASSES A CODE TO COPROCESSOR #UU AND RECEIVES A SET OF RETURN PARAMETERS TO BE LOADED INTO CPU REGISTERS
CONDITION-CODE:	UNAFFECTED

INSTRUCTION FORMAT:

15	14	13	12	11 10	9	8	7	6 5 4	3 2 1 0
0	1	0	0	UU	0	1	0	CNT	CODE

INSTRUCTION FIELDS:

UU FIELD–SPECIFIES HARDWARE BLOCK (COPROCESSOR)
- 00 – BLOCK 0
- 01 – BLOCK 1
- 10 – BLOCK 2
- 11 – BLOCK 3

CNT FIELD–SPECIFIES NUMBER OF REGISTERS TO PASS, BEGINNING WITH R4
- 000 – RESERVED, DO NOT USE
- 001 – PASS R4
- 010 – PASS R4–R5
- ⋮
- 111 – PASS R4–R10

FIG.23

H_EXEC	HARDWARE ACCELERATOR (COPROCESSOR) EXECUTE PRIMITIVE
OPERATION:	PASS EXECUTION CODE TO HARDWARE ACCELERATOR
ASSEMBLER SYNTAX:	H_EXEC #UU, #CODE
DESCRIPTION:	H_EXEC IS USED TO CONTROL A FUNCTION IN COPROCESSOR #UU. THE CODE FIELD IS NOT INTERPRETED BY THE CPU
CONDITION-CODE:	UNAFFECTED

INSTRUCTION FORMAT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		0	0	CODE							

INSTRUCTION FIELDS:

UU FIELD–SPECIFIES HARDWARE BLOCK (COPROCESSOR)
- 00 – BLOCK 0
- 01 – BLOCK 1
- 10 – BLOCK 2
- 11 – BLOCK 3

CODE FIELD–SPECIFIES AN OPERATION CODE FOR A HARDWARE BLOCK

FIG.24

H_LD HARDWARE ACCELERATOR (COPROCESSOR) LOAD PRIMITIVE

OPERATION: LOAD OPERAND FROM MEMORY AND PASS TO HARDWARE ACCELERATOR

ASSEMBLER SYNTAX:
H_LD.[HW][U] #UU, (RX, DISP)
H_LD.[U] #UU, (RX, DISP)

DESCRIPTION: H_LD PERFORMS A LOAD OF A VALUE IN MEMORY, AND PASSES THE MEMORY OPERAND TO THE COPROCESSOR WITHOUT STORING IT IN A GPR. THE H_LD OPERATION HAS THREE OPTIONS, W-WORD, H-HALF WORD AND U-UPDATE. DISP IS OBTAINED BY SCALING THE IMM2 FIELD BY THE SIZE OF THE LOAD, AND ZERO-EXTENDING. THIS VALUE IS ADDED TO THE VALUE OF REGISTER RX AND A LOAD OF THE SPECIFIED SIZE IS PERFORMED FROM THIS ADDRESS, WITH THE RESULT OF THE LOAD PASSED TO THE HARDWARE INTERFACE. FOR HALFWORD LOADS, THE DATA FETCHED IS ZERO-EXTENDED TO TO 32-BITS. IF THE .U OPTION IS SPECIFIED, THE EFFECTIVE ADDRESS OF THE LOAD IS PLACED IN REGISTER RX AFTER IT IS CALCULATED

CONDITION-CODE: UNAFFECTED

INSTRUCTION FORMAT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		1	SZ	0	UP	IMM2		RX			

INSTRUCTION FIELDS:

UU FIELD-SPECIFIES HARDWARE BLOCK (COPROCESSOR)
- 00 - BLOCK 0
- 01 - BLOCK 1
- 10 - BLOCK 2
- 11 - BLOCK 3

SIZE-SPECIFIES LOAD SIZE
- 0 - WORD
- 1 - HALFWORD

UP-SPECIFIES WHETHER THE BASE REGISTER SHOULD BE UPDATED
- 0 - NO UPDATE
- 1 - UPDATE BASE REGISTER WITH EFFECTIVE ADDRESS

IMM2 FIELD-SPECIFIES A 2-BIT SCALED IMMEDIATE VALUE

REGISTER X-SPECIFIES THE BASE ADDRESS TO BE ADDED TO THE SCALED IMMEDIATE FIELD

FIG.25

H_ST	HARDWARE ACCELERATOR (COPROCESSOR) STORE PRIMITIVE

OPERATION: STORE OPERAND TO MEMORY FROM HARDWARE ACCELERATOR

ASSEMBLER
SYNTAX: H_ST.[HW][U] #UU, (RX, DISP)

DESCRIPTION: H_ST PERFORMS A STORE TO MEMORY, OF AN OPERAND FROM A COPROCESSOR WITHOUT STORING IT IN A GPR. THE H_ST OPERATION HAS W-WORD, H-HALF WORD AND U-UPDATE. DISP IS OBTAINED BY SCALING THE IMM2 FIELD BY THE SIZE OF THE STORE AND ZERO-EXTENDING. THIS VALUE IS ADDED TO THE VALUE OF EGISTER RX AND STORE OF THE SPECIFIED SIZE IS PERFORMED TO THIS ADDRESS, WITH THE DATA FOR THE STORE OBTAINED FROM THE HARDWARE INTERFACE. IF THE .U OPTION IS SPECIFIED, THE EFFECTIVE ADDRESS OF THE LOAD IS PLACED IN REGISTER RX AFTER IT IS CALCULATED

CONDITION-CODE: UNAFFECTED

INSTRUCTION FORMAT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		1	SZ	1	UP	IMM2		RX			

INSTRUCTION FIELDS:

UU FIELD-SPECIFIES HARDWARE BLOCK (COPROCESSOR)
- 00 - BLOCK 0
- 01 - BLOCK 1
- 10 - BLOCK 2
- 11 - BLOCK 3

SIZE-SPECIFIES STORE SIZE
- 0 - WORD
- 1 - HALFWORD

UP-SPECIFIES WHETHER THE BASE REGISTER SHOULD BE UPDATED
- 0 - NO UPDATE
- 1 - UPDATE BASE REGISTER WITH EFFECTIVE ADDRESS

IMM2 FIELD-SPECIFIES A 2-BIT SCALED IMMEDIATE VALUE

REGISTER X-SPECIFIES THE BASE ADDRESS TO BE ADDED TO THE SCALED IMMEDIATE FIELD

FIG.26